Métodos Numéricos

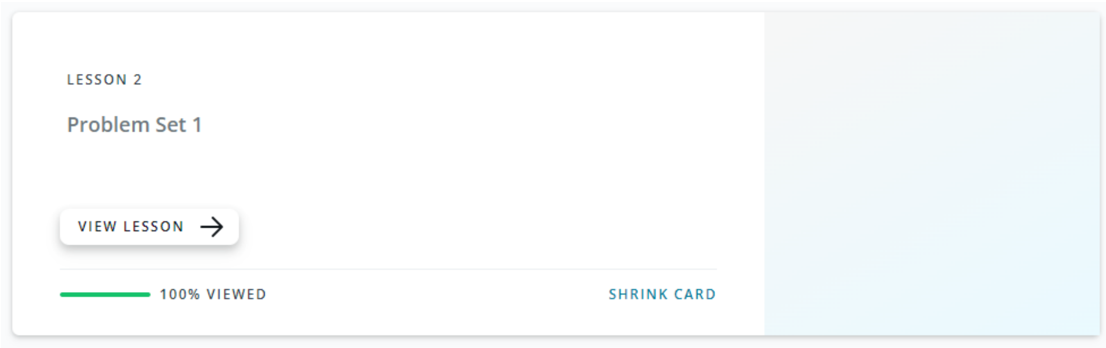
Mtro. Adolfo Centeno T

Differential Equations in Action - Lesson 2

Integrantes:

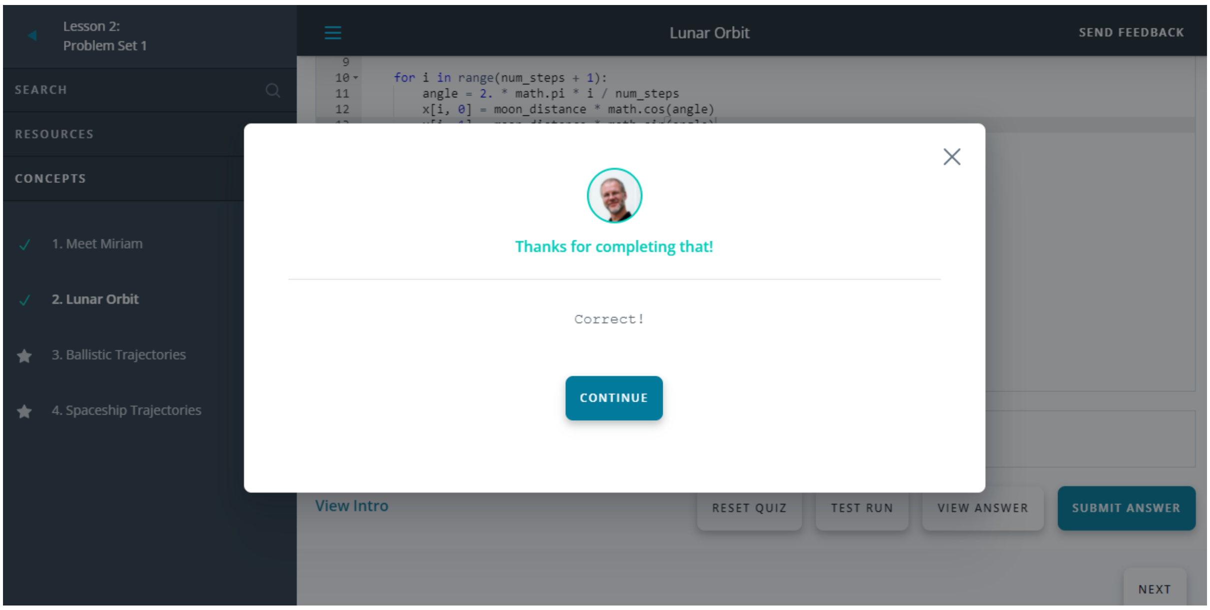
Maximiliano Macotela Nava, Luz Karen Hernández Hernández, Andres Serrato Barrera, Julio Rodríguez Salcedo, César David Rosales Álvares, Jim Kevin Holguín Rodríguez

Lesson 2 - Problem Set 1



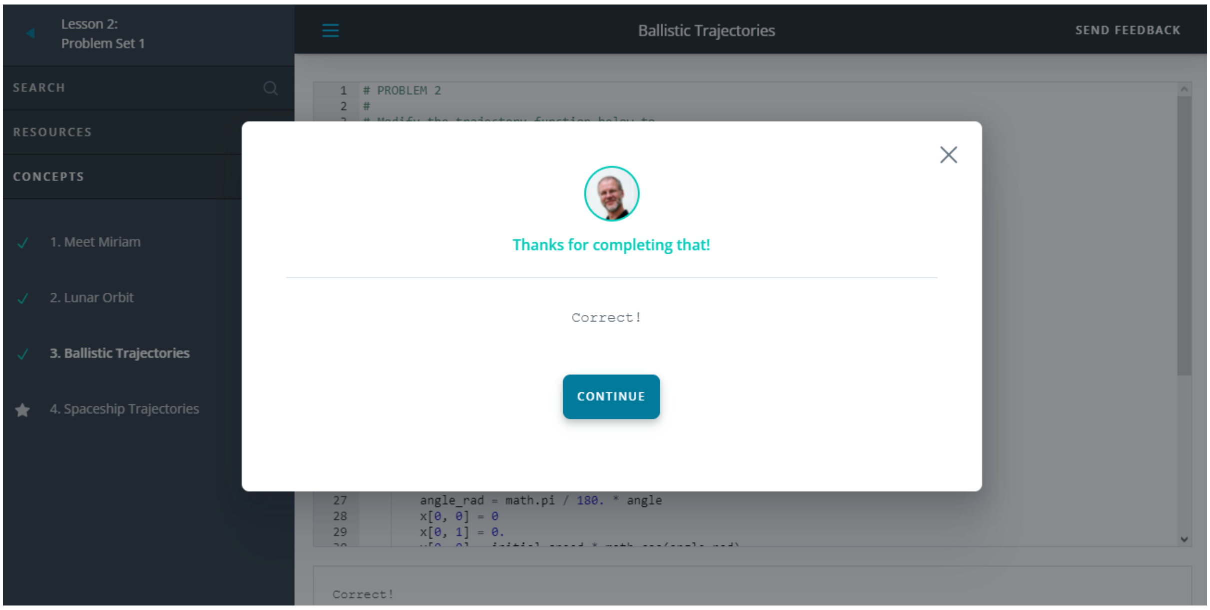
2. Lunar Orbit

```
import math
from udacityplots import *

moon_distance = 384e6 # m
def orbit():
    num_steps = 50
    x = numpy.zeros([num_steps + 1, 2])

    for i in range(num_steps + 1):
        angle = 2. * math.pi * i / num_steps
        x[i, 0] = moon_distance * math.cos(angle)
        x[i, 1] = moon_distance * math.sin(angle)
    return x
x = orbit()
@show_plot
def plot_me():
    matplotlib.pyplot.axis('equal')
    matplotlib.pyplot.plot(x[:, 0], x[:, 1])
    axes = matplotlib.pyplot.gca()
    axes.set_xlabel('Longitudinal position in m')
    axes.set_ylabel('Lateral position in m')
plot_me()
```



3. Ballistic Trajectories

```
import math
from udacityplots import *

h = 0.1 # s
g = 9.81 # m / s2
acceleration = numpy.array([0., -g])
initial_speed = 20. # m / s
@show_plot
def trajectory():
    angles = numpy.linspace(20., 70., 6)
    num_steps = 30
    x = numpy.zeros([num_steps + 1, 2])
    v = numpy.zeros([num_steps + 1, 2])
    for angle in angles:
        angle_rad = math.pi / 180. * angle
        x[0, 0] = 0
        x[0, 1] = 0.
        v[0, 0] = initial_speed * math.cos(angle_rad)
        v[0, 1] = initial_speed * math.sin(angle_rad)
        for step in range(num_steps):
            x[step + 1] = x[step] + h * v[step]
            v[step + 1] = v[step] + h * acceleration
        matplotlib.pyplot.plot(x[:, 0], x[:, 1])
    matplotlib.pyplot.axis('equal')
    axes = matplotlib.pyplot.gca()
    axes.set_xlabel('Horizontal position in m')
    axes.set_ylabel('Vertical position in m')
    return x, v
trajectory()
```



4. Spaceship Trajectories

```
from udacityplots import *

h = 1.0 # s
earth_mass = 5.97e24 # kg
gravitational_constant = 6.67e-11 # N m2 / kg2
def acceleration(spaceship_position):
    vector_to_earth = - spaceship_position
    return gravitational_constant * earth_mass /
numpy.linalg.norm(vector_to_earth)**3 * vector_to_earth
def ship_trajectory():
    num_steps = 13000
    x = numpy.zeros([num_steps + 1, 2]) # m
    v = numpy.zeros([num_steps + 1, 2]) # m / s
    x[0, 0] = 15e6
    x[0, 1] = 1e6
    v[0, 0] = 2e3
    v[0, 1] = 4e3
    for step in range(num_steps):
        x[step + 1] = x[step] + h * v[step]
        v[step + 1] = v[step] + h * acceleration(x[step])
    return x, v
x, v = ship_trajectory()
@show_plot
def plot_me():
    matplotlib.pyplot.plot(x[:, 0], x[:, 1])
    matplotlib.pyplot.scatter(0, 0)
    matplotlib.pyplot.axis('equal')
    axes = matplotlib.pyplot.gca()
    axes.set_xlabel('Longitudinal position in m')
    axes.set_ylabel('Lateral position in m')
plot_me()
```